Rest

Corporación Universitaria Adventista

Ingeniería de Sistemas

Programación Web

Zamirt Barrera

Sogamoso, Colombia

2021

REST

REST es una tecnología mucho más flexible que transporta datos por medio del protocolo HTTP, pero este permite utilizar los diversos métodos que proporciona HTTP para comunicarse, como lo son GET, POST, PUT, DELETE, PATCH y a la vez, utiliza los códigos de respuesta nativos de HTTP (404,200,204,409).

REST es tan flexible que permite transmitir prácticamente ***cualquier tipo de datos***, ya que el tipo de datos está definido por el Header Content-Type, lo que nos permite mandar, XML, JSON, Binarios (imágenes, documentos), Textos, etc.

El acrónimo REST se refiere al estado del recurso al que accede la API, no al estado de una sesión dentro de la cual se llama a la API. Si bien puede haber buenas razones para crear una API con estado, es importante darse cuenta de que administrar sesiones es complejo y difícil de hacer de forma segura.

API REST

La API REST utiliza HTTP para recopilar datos, transferir datos y coordinar la ejecución de tareas entre sistemas remotos.

Representational State Transfer (REST) es un enfoque de implementación de API que utiliza el Protocolo de transferencia de hipertexto (HTTP) para crear conexiones.

Las API REST no tienen estado. Para completar una conexión, una API REST no necesita el cliente o el servidor.

SEGURIDAD REST

Las API REST usan el Protocolo seguro de transferencia de hipertexto (HTTPS).

Un protocolo de Seguridad de la capa de transporte (TLS) garantiza que la conexión sea privada (mediante el cifrado de los datos), autenticada (mediante criptografía de clave pública) y confiable (mediante el uso de un código de autenticación de mensaje).

La solicitud HTTP se encarga de eso al recopilar y guardar la información solicitada. Esto significa que si los actores de la amenaza obtienen acceso a la solicitud HTTP, obtienen acceso a los datos.

SEGURIDAD OWASP REST

Owasp rest es un documento que contiene las mejores prácticas para asegurar la API REST. Cada sección aborda un componente dentro de la arquitectura REST y explica cómo debe lograrse de manera segura.

Fielding escribió las especificaciones HTTP / 1.1 y URI y se ha demostrado que es adecuado para desarrollar aplicaciones hipermedia distribuidas. Si bien REST tiene una aplicación más amplia, se usa con mayor frecuencia en el contexto de la comunicación con servicios a través de HTTP.

AUTENTICACIÓN API REST

Hay muchos métodos de seguridad y muchas formas de autentificarse, que pueden depender del tipo de dispositivo, el tipo de uso, la confidencialidad de la información, entre otros. No hay una sola manera de asegurar tu API.

Autenticación básica

Esta es la forma más sencilla de asegurar tu API. Se basa principalmente en un nombre de usuario y una contraseña para identificarte.

Para comunicar estas credenciales desde el cliente hasta el servidor, se debe realizar mediante el encabezado HTTP Autorización (Authorization), según la especificación del protocolo HTTP.

Este método de autentificación está algo anticuado y puede ser un problema de seguridad en tu API REST.

Cualquiera que intercepte la trasmisión de datos puede decodificar fácilmente esta información. Esto se denomina ataque Man-In-The-Middle (MiTM).

Para proteger tu API mediante la autenticación básica debes configurar que las conexiones entre los clientes y tu servicio API funcionen únicamente mediante una conexión TLS/HTTPS, nunca sobre HTTP.

Autentificación basada en token

el usuario se identifica al igual que con la autenticación básica, con sus credenciales, nombre de usuario y contraseña. Pero en este caso, con la primera petición de autentificación, el servidor generará un token basado en esas credenciales.

Por norma general, los tokens están codificados con la fecha y la hora para que en caso de que alguien intercepte el token con un ataque MiTM, no pueda utilizarlo pasado un tiempo establecido. Además de que el token se puede configurar para que caduque después de un tiempo definido, por lo que los usuarios deberán iniciar sesión de nuevo.

Autenticación basada en clave API

debes configurar el acceso a los recursos de tu API. Tu sistema API debe generar una clave (*key*) y un *secret key* para cada cliente que requiera acceso a tus servicios. Cada vez que una aplicación necesite consumir los datos de tu API, deberás enviar tanto la *key* como la *secret key*.

Este sistema es más seguro que los métodos anteriores, pero la generación de credenciales debe ser manual y esto dificulta la escalabilidad de tu API. La automatización de generación e intercambio de *key's* es una de las razones principales por las que se desarrolló el método de autentificación OAuth, que en el siguiente punto evaluaremos.

Otros problemas con la autenticación basada en clave API es la administración de claves. Con tareas tan relevantes como:

a. Genera la *key* y el *secret key.*

b. Enviar las credenciales a los desarrolladores.

c. Guardar de forma segura la *key* y el *secret key.*

OAuth 2.0

OAuth 2.0 es un método de autorización utilizado por compañías como Google, Facebook, Twitter, Amazon, Microsoft, etc. Su propósito es permitir a otros proveedores, servicios o aplicaciones, el acceso a la información sin facilitar directamente las credenciales de los usuarios. Pero tranquilo, únicamente accederán bajo la confirmación del usuario, validando la información a la que se le autorizará acceder.

La protección y el control de la información de tus servicios es un requisito obligatorio a la hora de hacer públicos tus servicios mediante una API o cualquier otro sistema.

Aplicando el estándar solo HTTPS

HTTPS se habilita mediante el uso de certificados SSL (Secure Socket Layer). El primer paso es comprar un certificado de su proveedor de host. A continuación, deberá instalar el certificado. Esto se realiza a través del panel de hosting, y es tan simple como seguir los pasos proporcionados por el host.

Hacer que su sitio sea solo HTTPS significa revisar sus directorios (como bibliotecas de clientes, ejemplos de código y aplicaciones de muestra) y reemplazar las llamadas de HTTP con HTTPS. Puede hacerlo manualmente, lo que lleva tiempo, o puede usar una herramienta gratuita de búsqueda y reemplazo .

Uso de un proveedor de identidad (IdP) para tokens

Un IdP es un sistema que controla el proceso de creación y gestión de información de identidad y proporciona servicios de autenticación como la generación de tokens. Una ficha es una secuencia de caracteres sin sentido que reemplaza la información sensible y financiera.

El IdP recibe una solicitud del sitio web para tokenizar la información. Luego, el sistema guarda la información en una ubicación segura y la reemplaza con un token. El IdP media entre el usuario y el sitio web, manteniendo la información segura en un repositorio de terceros.

Firmas criptográficas para JWT

Las firmas digitales proporcionan integridad, autenticación y no repudio, lo que las hace ideales para emitir tokens. Puede delegar la tarea de generar tokens a un IdP (como se explicó anteriormente) o puede hacerlo manualmente mediante el uso de un servidor OAuth 2.0 verificado.

Puede construir su propio servidor OAuth, usar el servidor OAuth patrocinado por OKTA y puede usar una solución de código abierto. Antes de comprometerse con un camino, evalúe su situación con honestidad y asegúrese de tener las habilidades y los recursos para hacerlo por su cuenta.

Retorno 429 HTTP para demasiadas solicitudes

El propósito del código de retorno 429 es evitar repetidas solicitudes de API. Depende de usted determinar cuántas solicitudes son demasiadas en un momento dado, pero este código de retorno es obligatorio. Le proporciona un mecanismo para proporcionar ataques DDoS, que podrían causar una interrupción del sistema.